

CocoaHeads

Dnepropetrovsk #1



Индексация пространственных данных (spatial indexing)

План

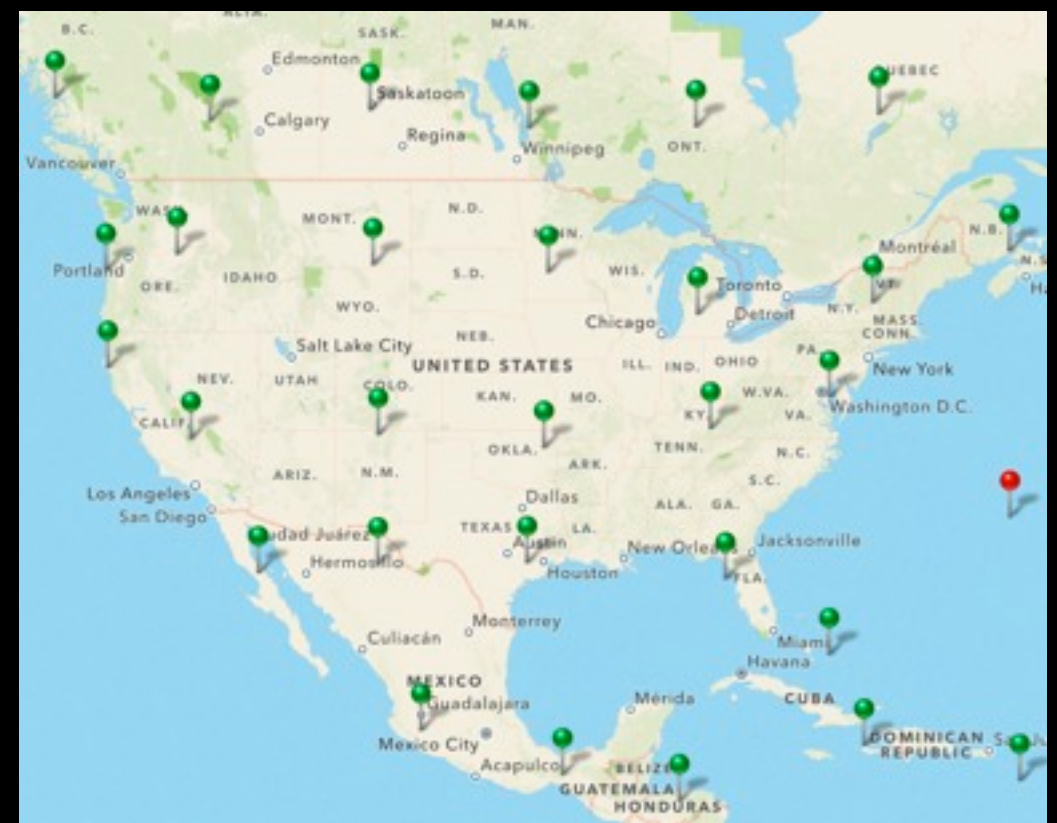
1. Суть проблемы
2. Решение Apple. Плюсы/Минусы
3. Суть выбранного решения.
4. Оптимизация
5. Демо
6. Выводы
7. Что ещё?

Зачем же нужна
кластеризация?

Зачем же нужна кластеризация?



Зачем же нужна кластеризация?



Варианты

- Grid
- Quadtree
- R-Tree
- R+ Tree
- R* Tree
- kd-tree

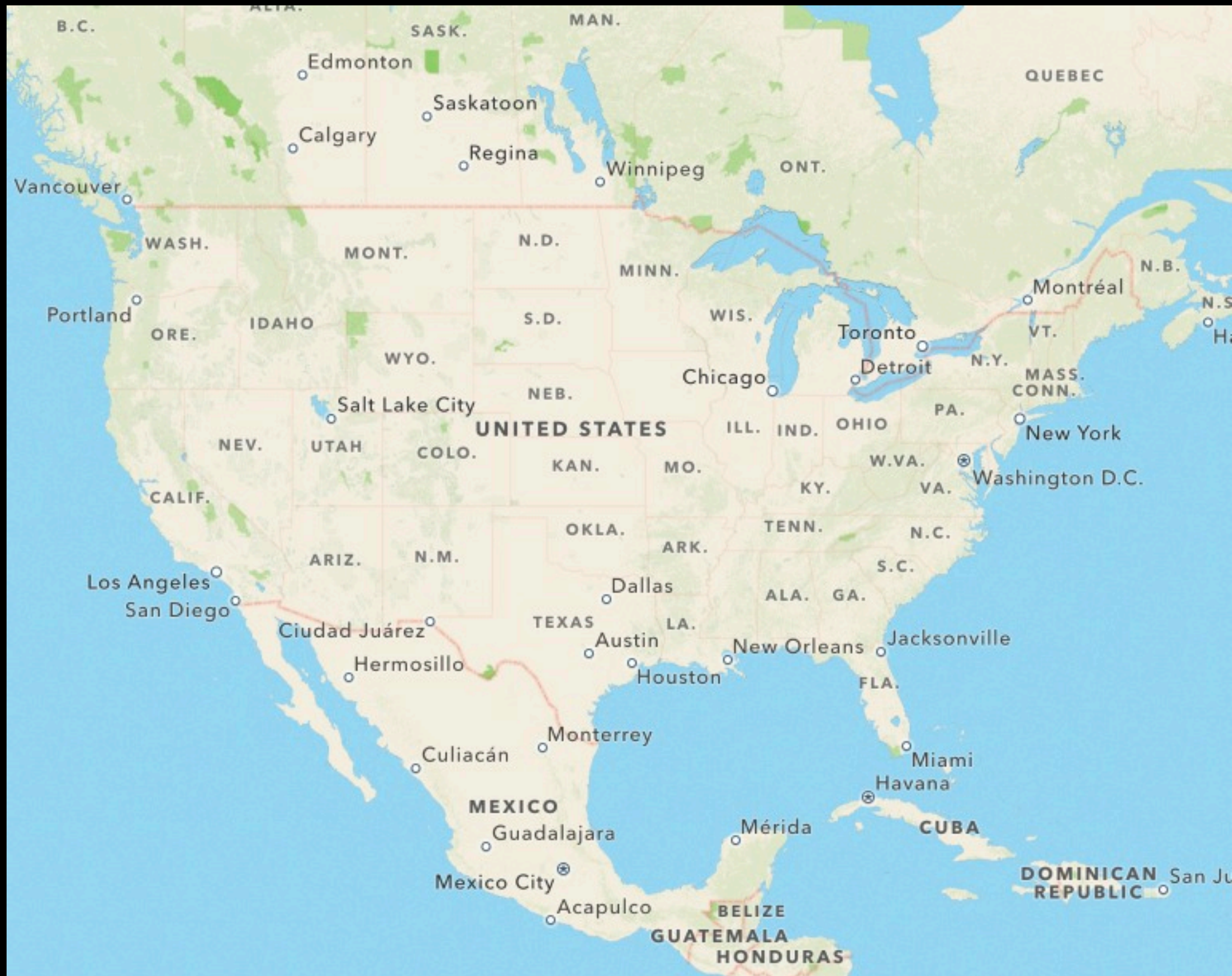
Что предлагает Apple:

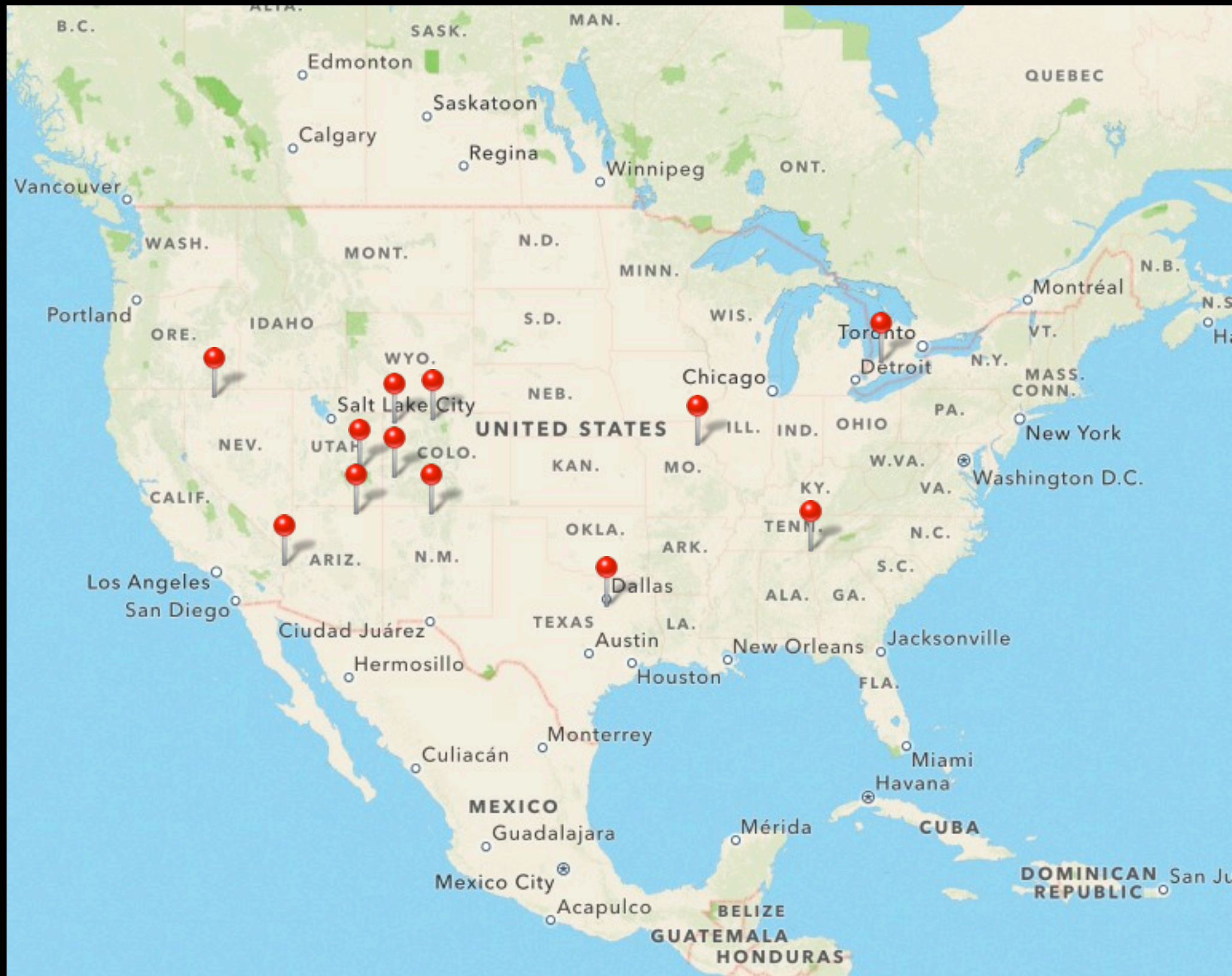
- WWDC'11 - Session 111 - Visualizing Geographically with MapKit - Grid

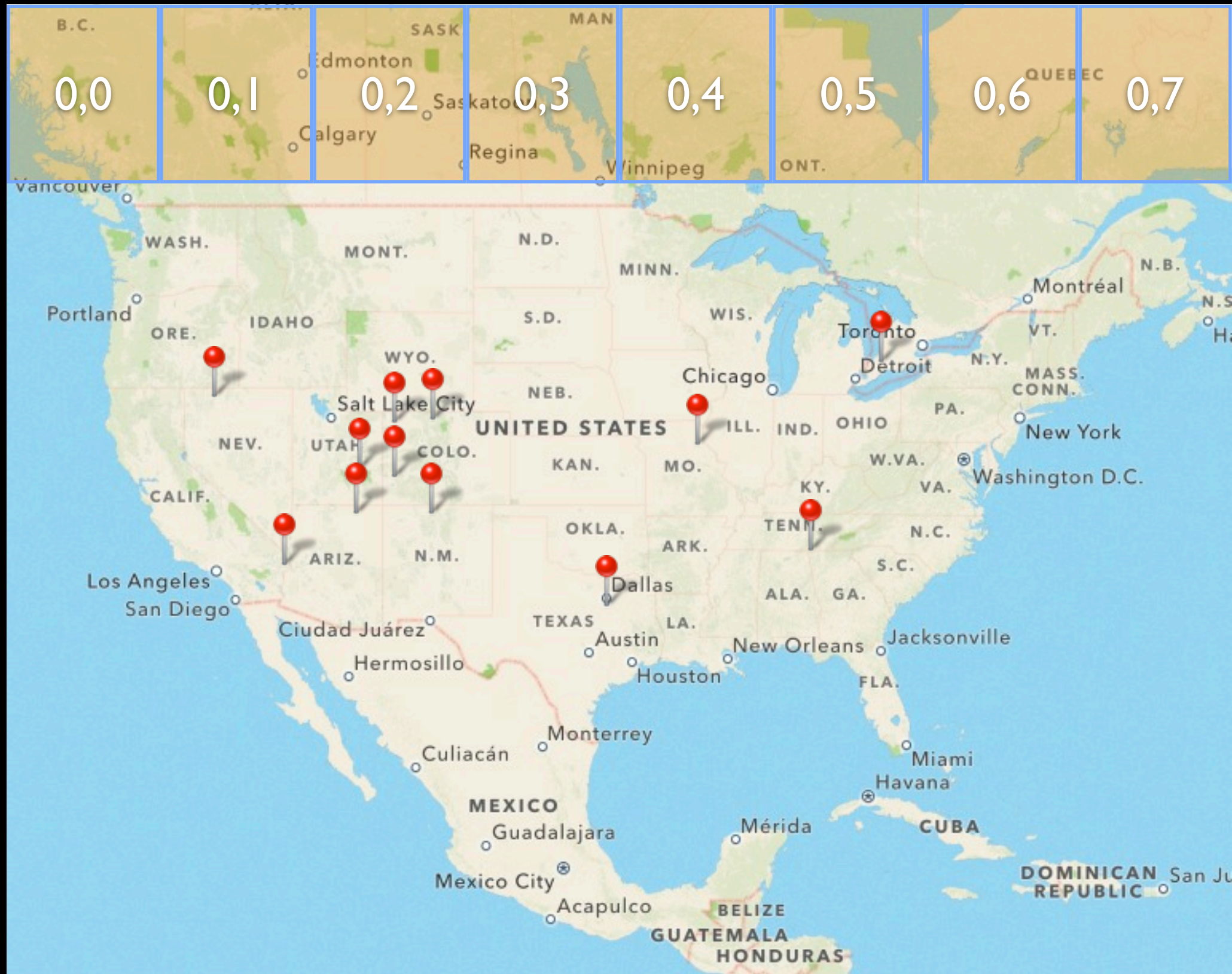
Grid

как это работает

- при каждом `mapView:regionDidChange:`
- для прямоугольника заданного размера находят ближайшую к центру точку









Что скажет Time Profiler?

- iPhone 4s, iOS 6.1, XCode 5, 34k объектов
- Добавление: ~900+ ms
- Кластеризация: ~2200+ ms

Варианты

- Grid - Apple
- Quadtree
- R-Tree
- R+ Tree
- R* Tree
- kd-tree

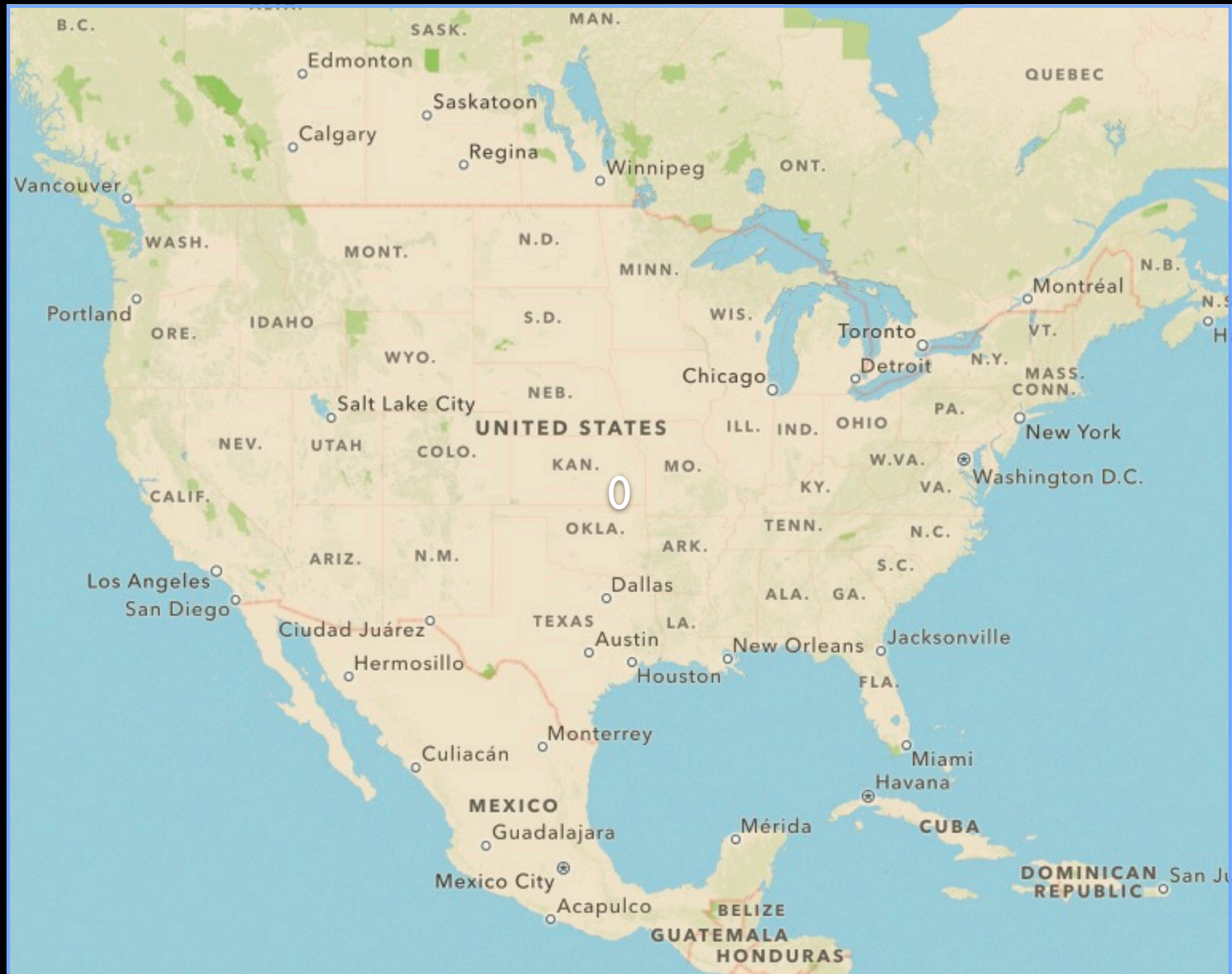
Почему Quadtree?

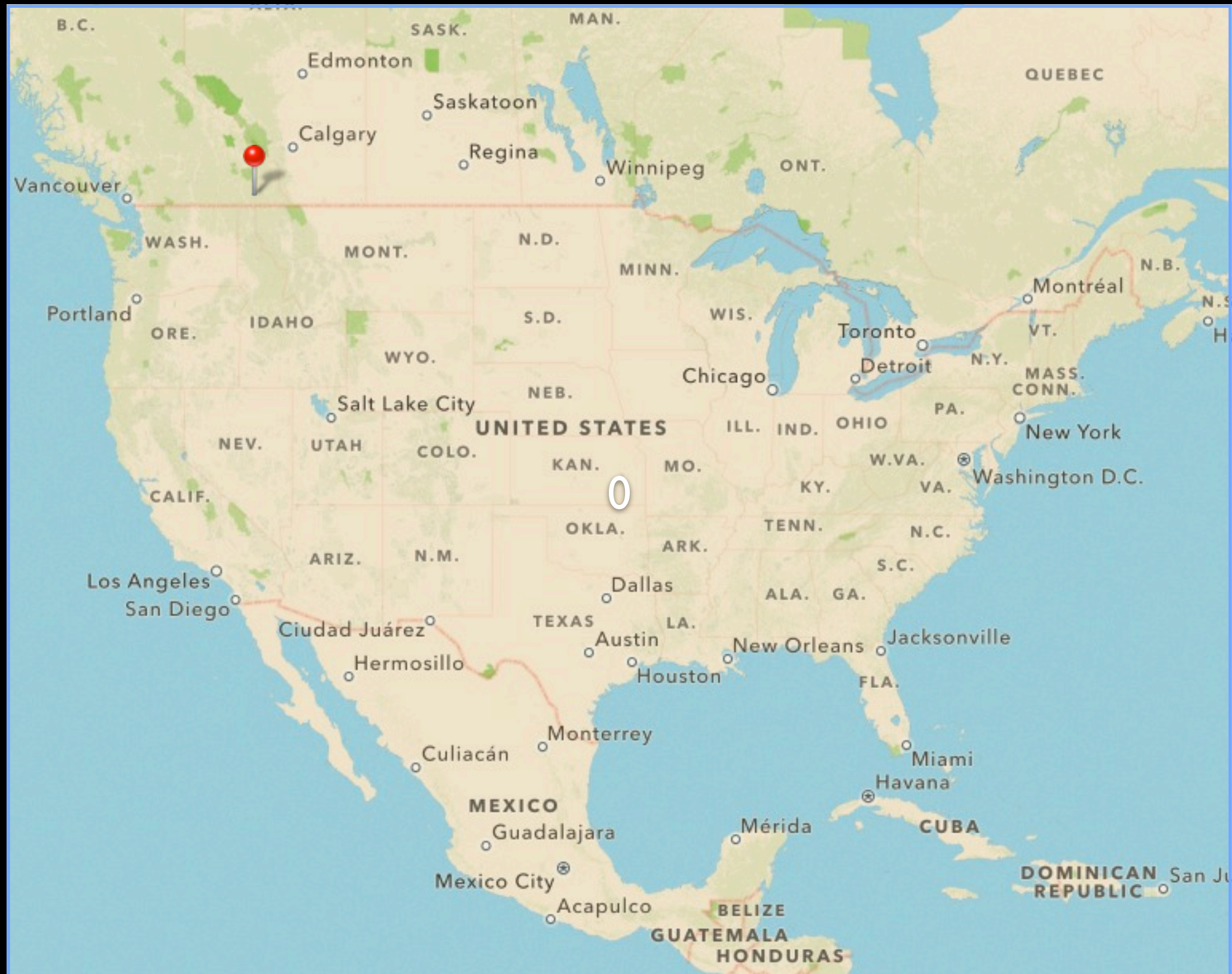
- Реализация Google Maps
- SuperPin
- Не надо решать вопросы балансировки (как в случае с kd-tree)
- Не знал о R-Tree

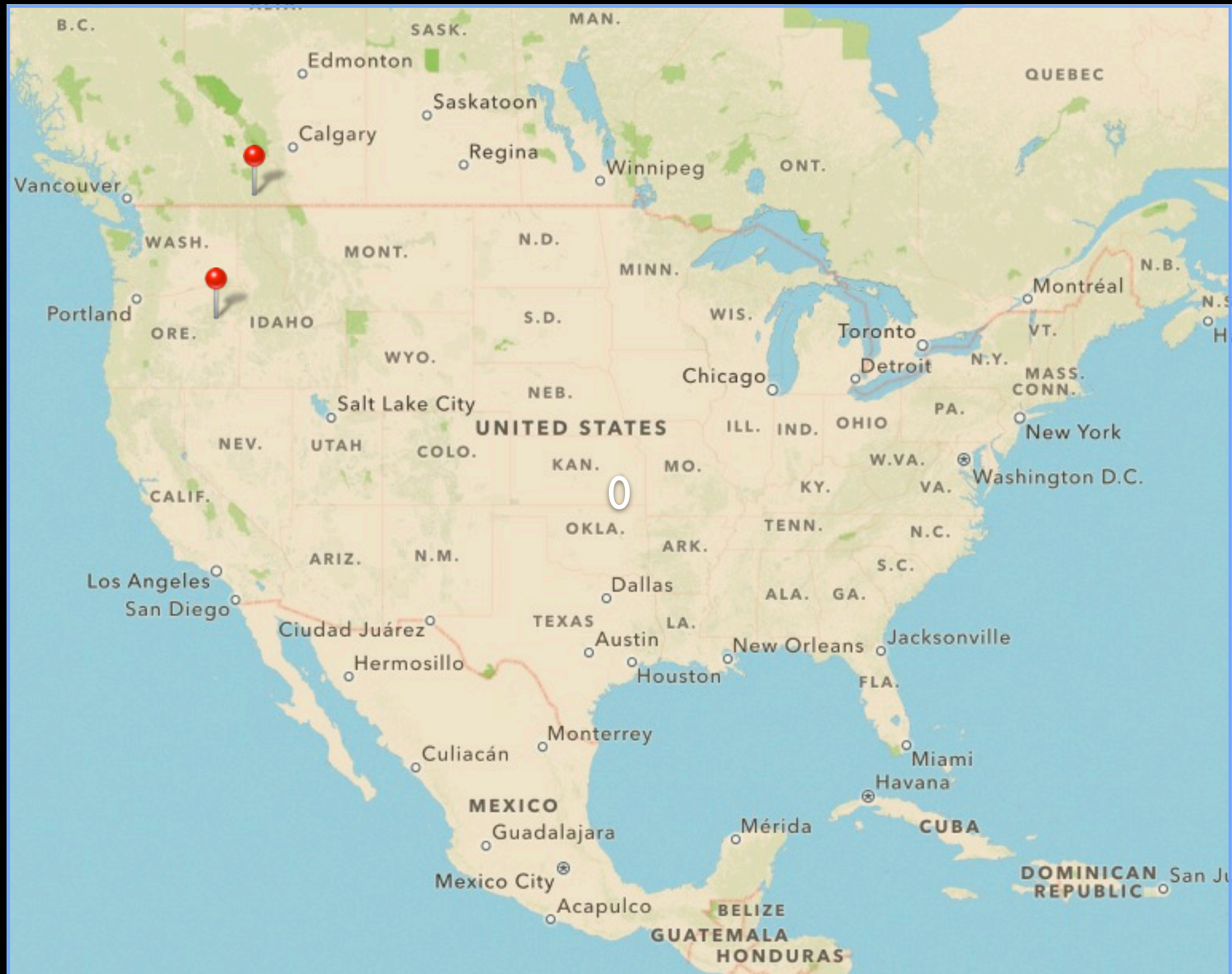
Quadtree

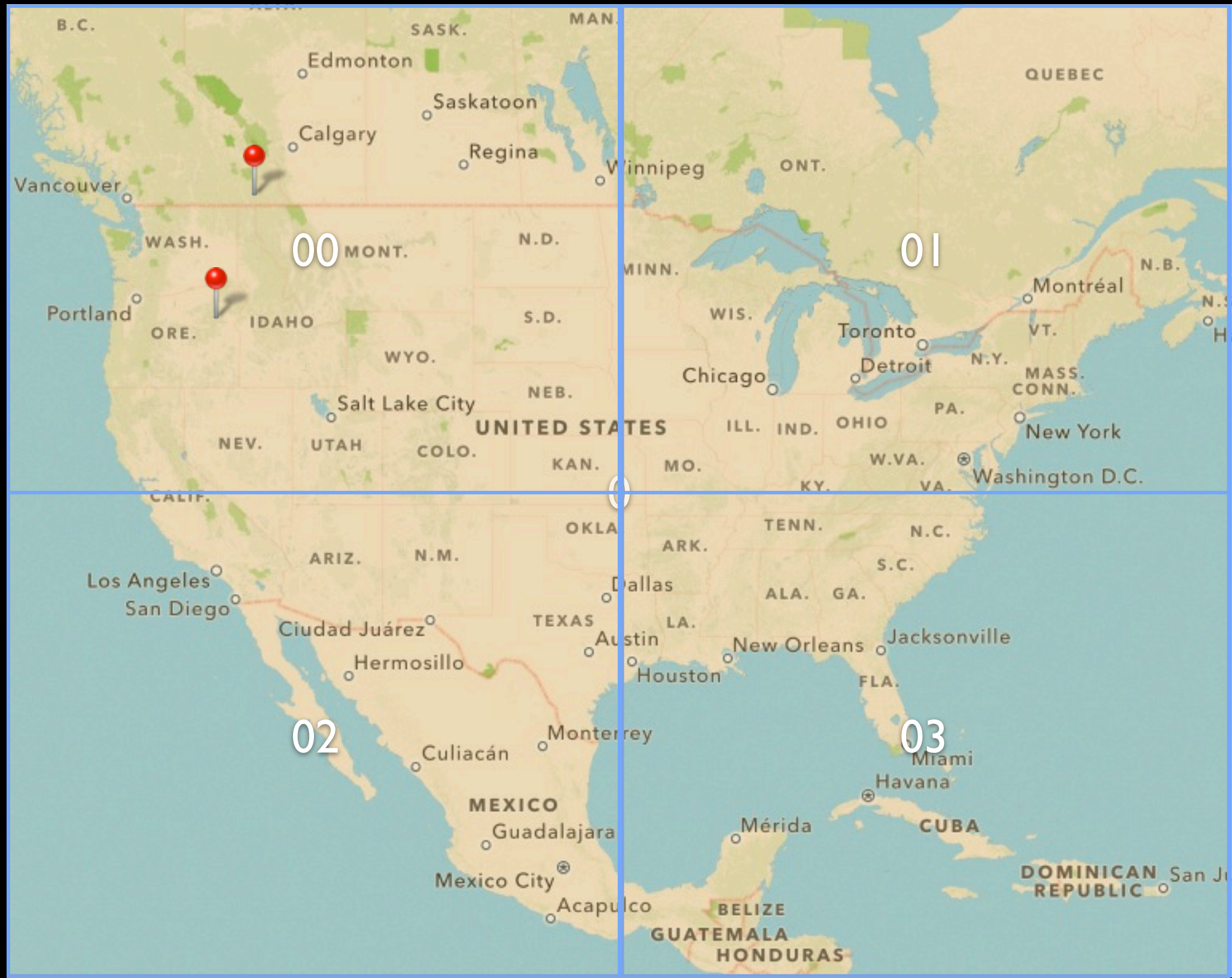
как это работает

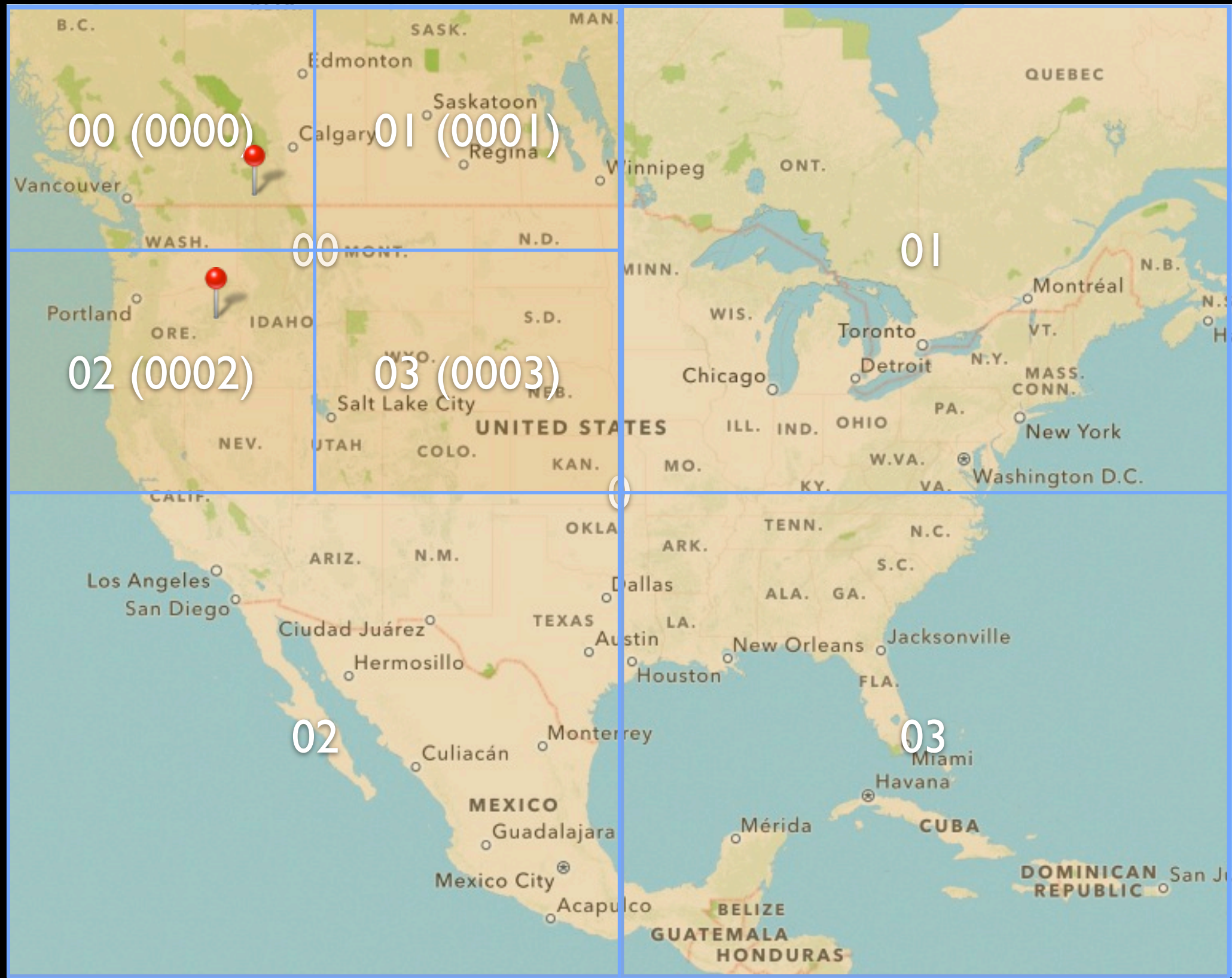
- Кластеры создаются путём выборки из индекса
- Расчёт центра масс для заданного квадранта при изменении индекса



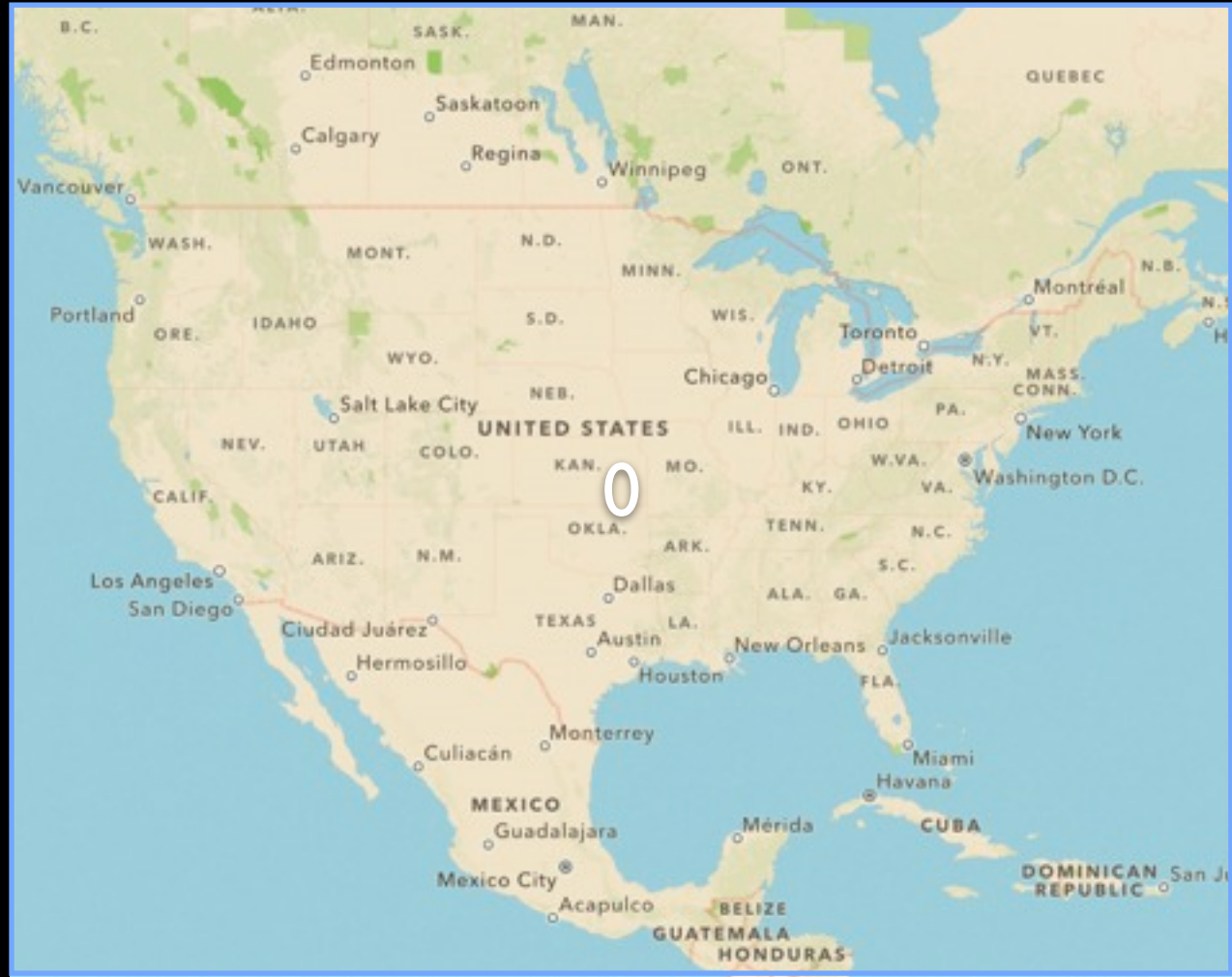


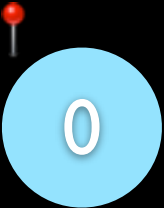
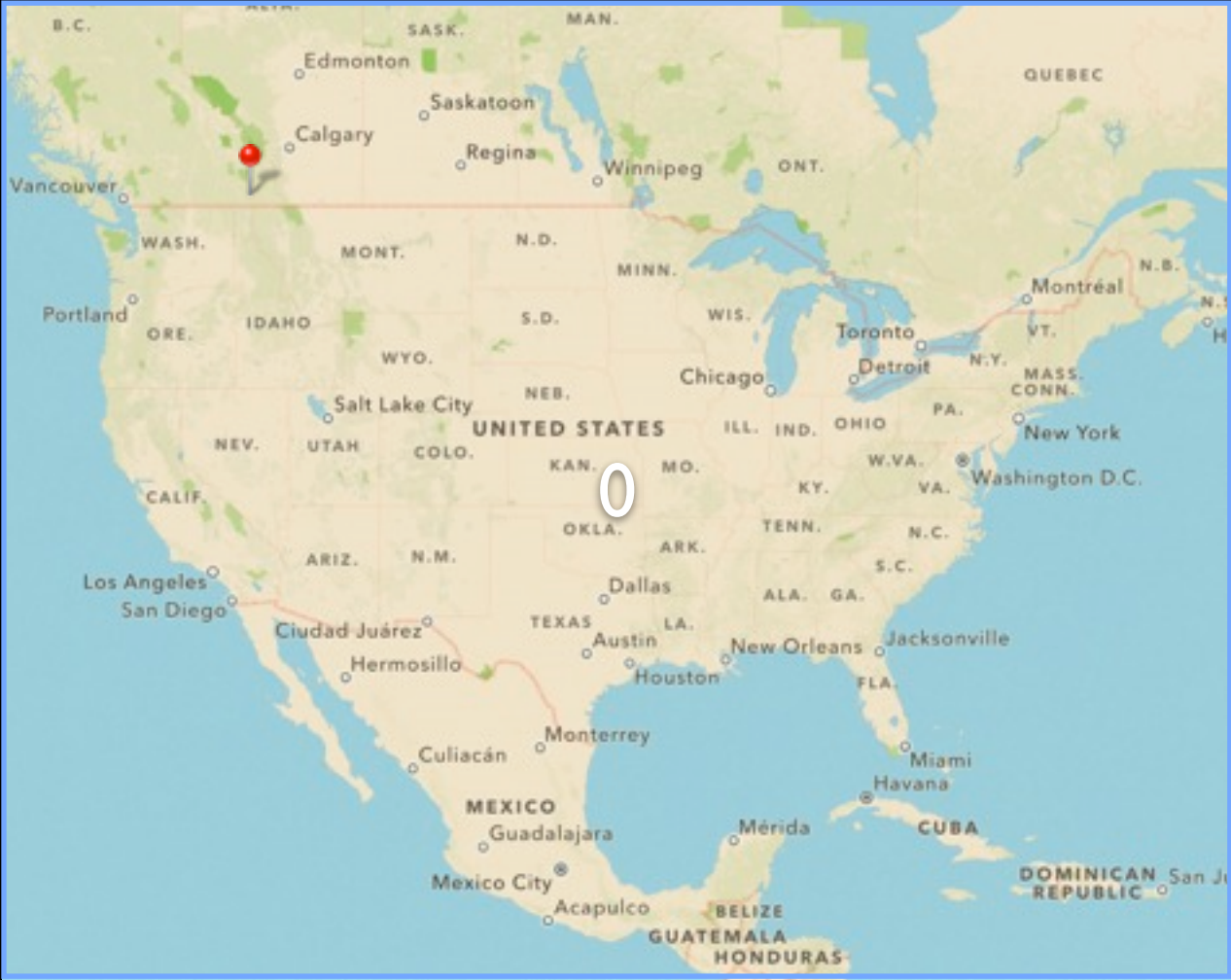


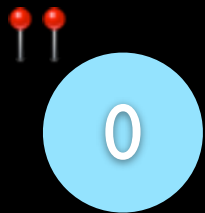
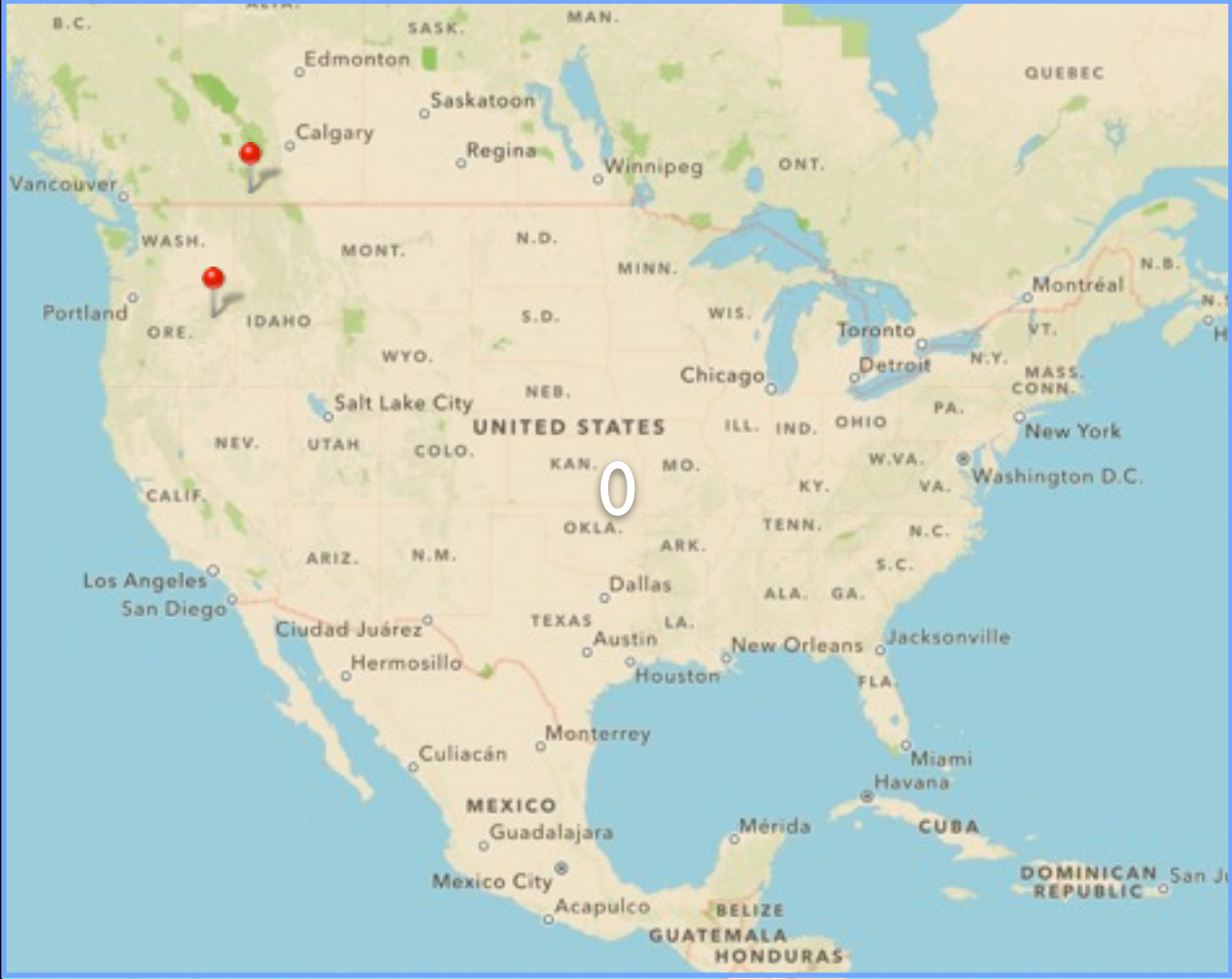


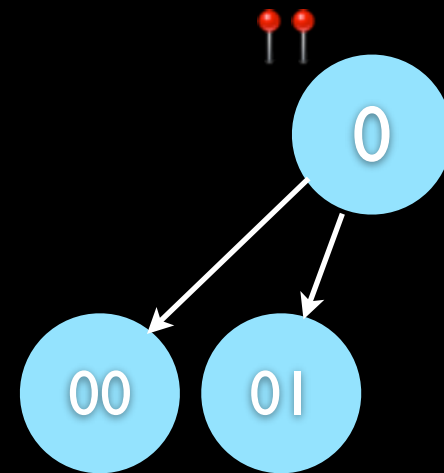
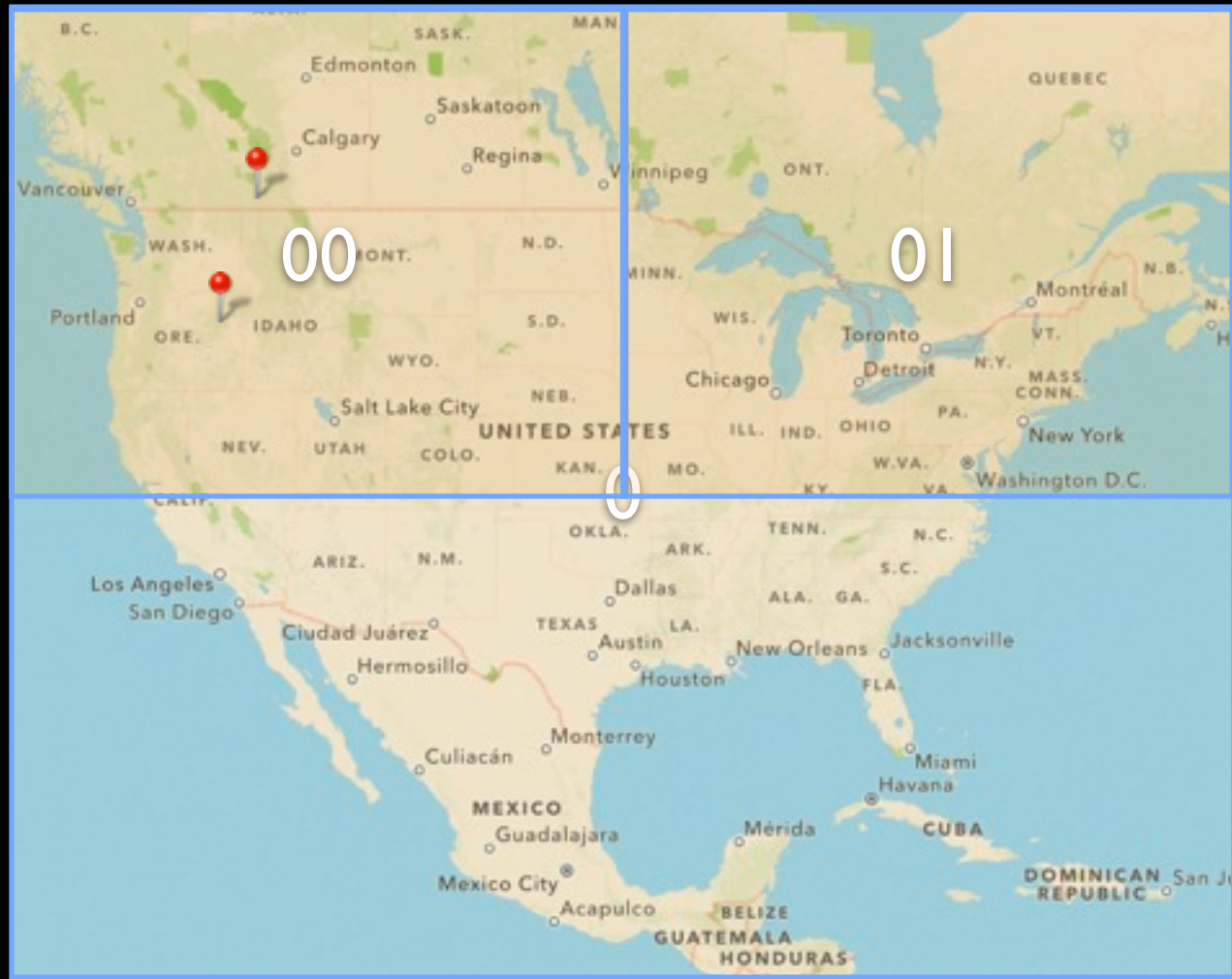


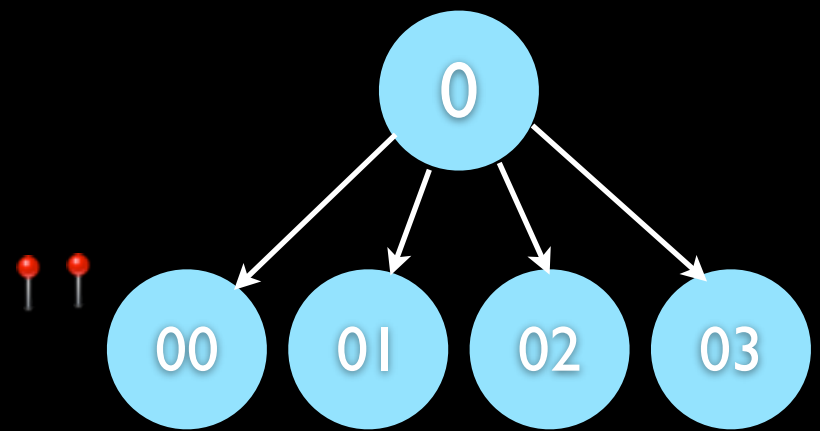
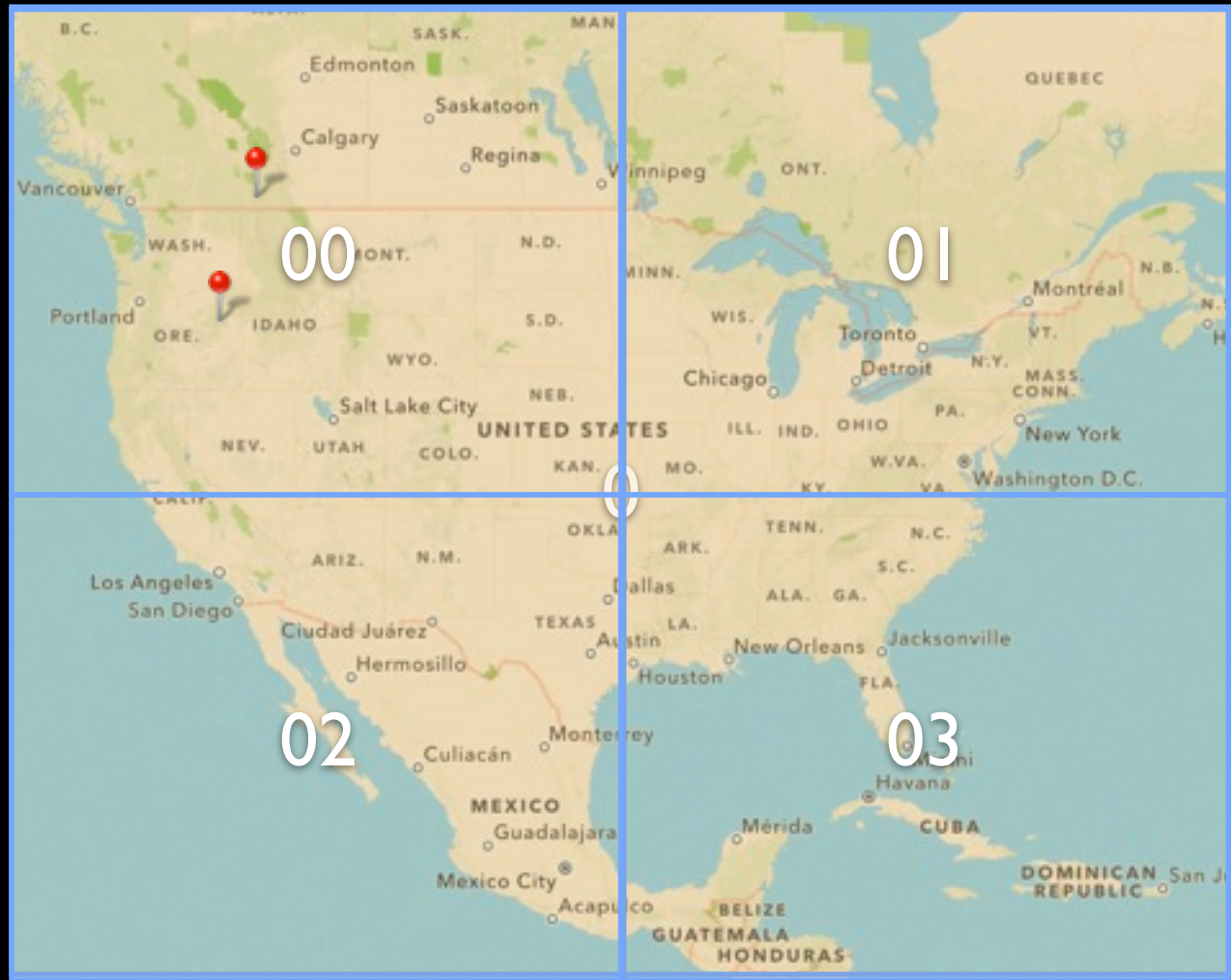
Повернём боком :)

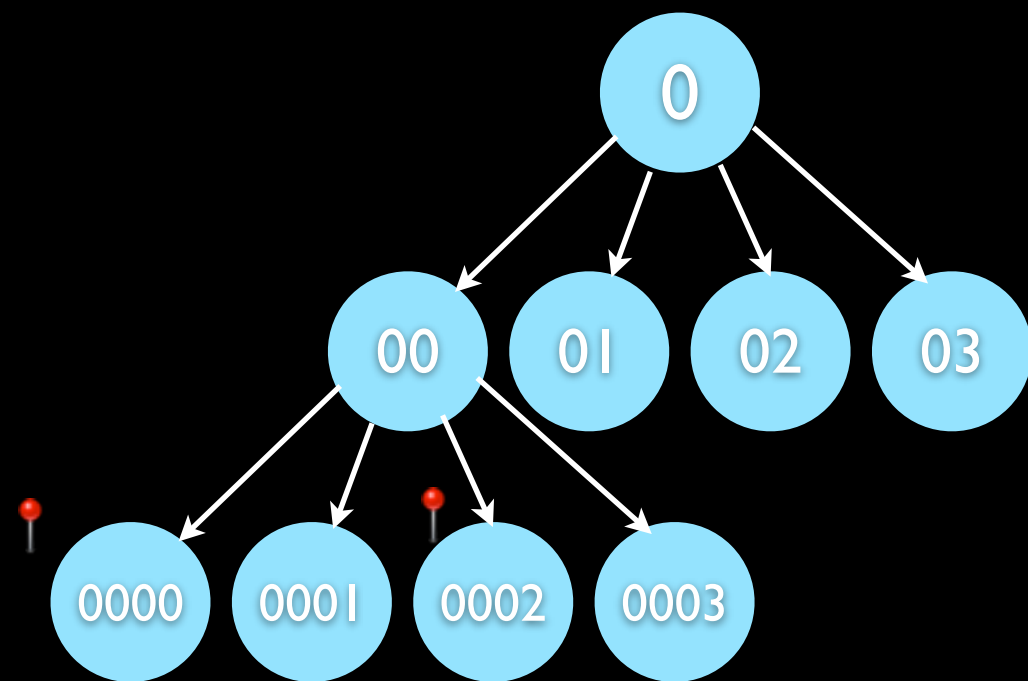
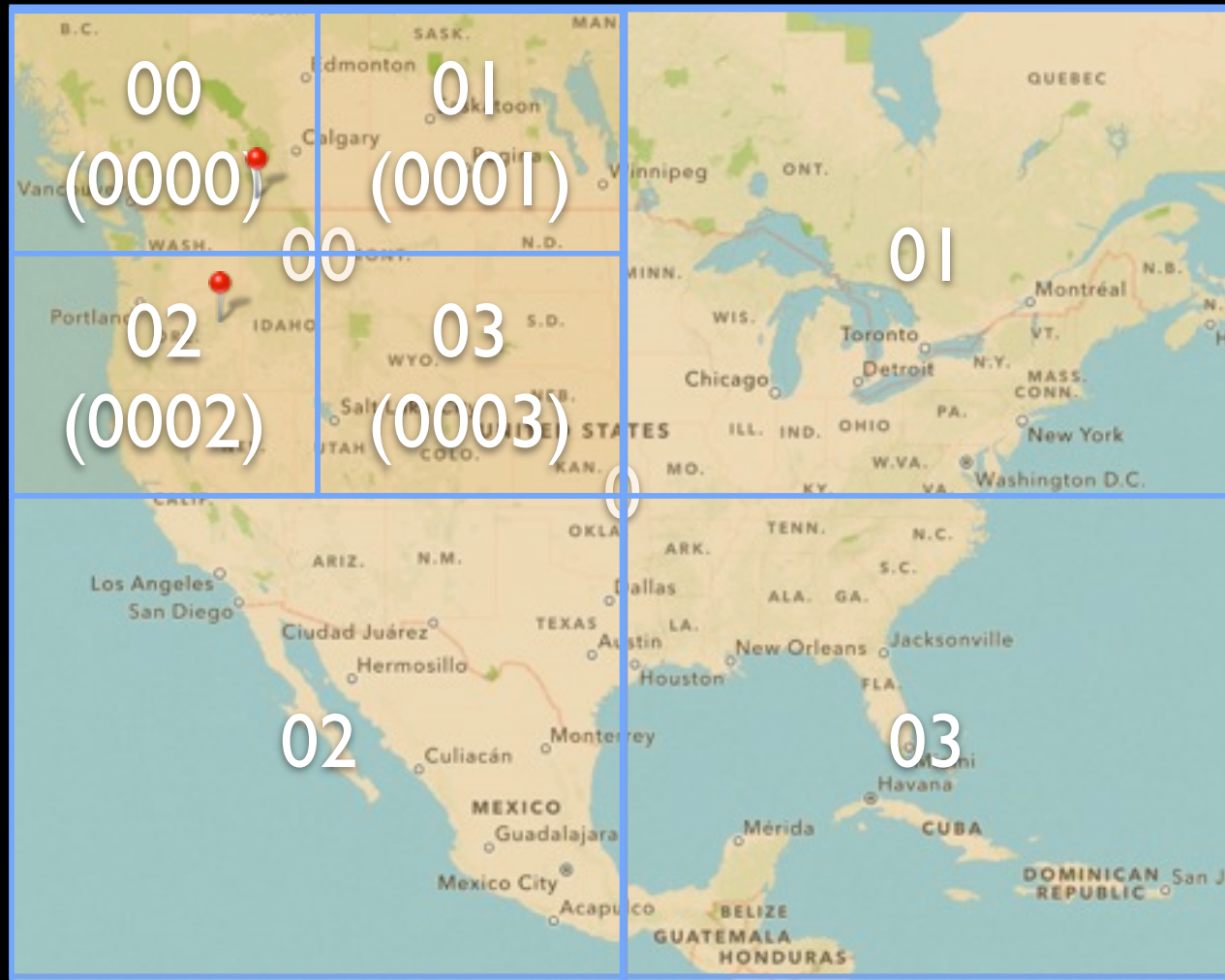












PR Quadtree кратко

- Разбиение пространства на равные квадранты
- 4 дочерних вершины
- Все POIs находятся в листовых вершинах (снизу)
- Листовая вершина не содержит более одной POI - Point-Region QuadTree
- При вставке второй POI происходит разбиение вершины и «передача» POIs дочерним вершинам

MIQuadTree

```
typedef struct MIQuadTree *MIQuadTreeRef;
```

```
struct MIQuadTree  
{  
    MIQuadTreeRef topLeftLeaf;  
    MIQuadTreeRef topRightLeaf;  
    MIQuadTreeRef bottomLeftLeaf;  
    MIQuadTreeRef bottomRightLeaf;  
  
    MKMapPoint centroid;  
    MKMapRect rect;  
  
    MIPointListRef pointList;  
    unsigned int count;  
  
    unsigned int level;  
    MIQuadTreeRef root;  
};
```

Связь с Obj-C

```
@interface MIMapIndex : NSObject
{
    MIQuadTreeRef _tree;
    NSMutableSet *_annotations;
}
```

- (void)addAnnotations:(NSArray *)annotations;
- (void)removeAnnotations:(NSArray *)annotations;
- (NSArray *)annotations;
- (NSSet *)annotationsInMapRect:(MKMapRect)mapRect;
- (void)annotationsInMapRect:(MKMapRect)mapRect
level:(NSUInteger)level
clusters:(NSMutableSet **)
points:(NSMutableSet **)points;

Вернёмся к выборке

Почему так быстро

```
typedef void (*MITraverseCallback)(MIPoint point, MITraverseResultType resultType, MITraverse *ctx)

void MIQuadTreeTraversLevelRectPoints(MIQuadTreeRef tree, MKMapRect rect, unsigned int level,
    MITraverse *traverse)
{
    if (tree->count == 0 || !MKMapRectIntersectsRect(tree->rect, rect)) return;

    if (tree->topLeftLeaf != NULL && tree->level < level) {
        ... // обход дочерних вершин
        return;
    }

    if (tree->count > 1)
    {
        traverse->callback(point, MITraverseResultTree, traverse);
    }
    else if (MKMapRectContainsPoint(rect, tree->pointList->point))
    {
        traverse->callback(tree->pointList->point, MITraverseResultPoint, traverse);
    }
}
```

Почему так быстро

```
typedef void (*MITraverseCallback)(MIPoint point, MITraverseResultType resultType, MITraverse *ctx)

void MIQuadTreeTraversLevelRectPoints(MIQuadTreeRef tree, MKMapRect rect, unsigned int level,
    MITraverse *traverse)
{
    if (tree->count == 0 || !MKMapRectIntersectsRect(tree->rect, rect)) return;

    if (tree->topLeftLeaf != NULL && tree->level < level) {
        ... // обход дочерних вершин
        return;
    }

    if (tree->count > 1)
    {
        traverse->callback(point, MITraverseResultTree, traverse);
    }
    else if (MKMapRectContainsPoint(rect, tree->pointList->point))
    {
        traverse->callback(tree->pointList->point, MITraverseResultPoint, traverse);
    }
}
```


Почему так быстро

```
typedef void (*MITraverseCallback)(MIPoint point, MITraverseResultType resultType, MITraverse *ctx)

void MIQuadTreeTraversLevelRectPoints(MIQuadTreeRef tree, MKMapRect rect, unsigned int level,
    MITraverse *traverse)
{
    if (tree->count == 0 || !MKMapRectIntersectsRect(tree->rect, rect)) return;

    if (tree->topLeftLeaf != NULL && tree->level < level) {
        ... // обход дочерних вершин
        return;
    }

    if (tree->count > 1)
    {
        traverse->callback(point, MITraverseResultTree, traverse);
    }
    else if (MKMapRectContainsPoint(rect, tree->pointList->point))
    {
        traverse->callback(tree->pointList->point, MITraverseResultPoint, traverse);
    }
}
```

Почему так быстро

```
typedef void (*MITraverseCallback)(MIPoint point, MITraverseResultType resultType, MITraverse *ctx)

void MIQuadTreeTraversLevelRectPoints(MIQuadTreeRef tree, MKMapRect rect, unsigned int level,
    MITraverse *traverse)
{
    if (tree->count == 0 || !MKMapRectIntersectsRect(tree->rect, rect)) return;

    if (tree->topLeftLeaf != NULL && tree->level < level) {
        ... // обход дочерних вершин
        return;
    }

    if (tree->count > 1)
    {
        traverse->callback(point, MITraverseResultTree, traverse);
    }
    else if (MKMapRectContainsPoint(rect, tree->pointList->point))
    {
        traverse->callback(tree->pointList->point, MITraverseResultPoint, traverse);
    }
}
```



Немного о вставке:

- Уникальность объектов и memory management обеспечивает NSMutableSet

```
[annotations enumerateObjectsUsingBlock:^(id<MKAnnotation> annotation...
{
    NSUInteger countBefore = _annotations.count;
    [_annotations addObject:annotation];
    if (countBefore < _annotations.count)
    {
        MIQuadTreeInsertPoint
        (
            _tree,
            MKMapPointForCoordinate([annotation coordinate]),
            (__bridge void *)annotation
        );
    }
}];
```

Что теперь скажет Time Profiler?

- Добавление: ~395 ms

361.0ms	8.2%	12,0		▼-[MIMapIndex addAnnotations:] Demo
349.0ms	7.9%	115,0		►__29-[MIMapIndex addAnnotations:]_block_invoke Demo

- Выборка в произвольном месте - ~1 ms

Лучше, чем ничего,
НО..

Затея №1

Заменим:

[annotations **enumerateObjectsUsingBlock:**^(id<MKAnnotation> annotation..

На:

[annotations **enumerate..Block:**^(__unsafe_unretained id<MKAnnotation>..)

Затея №1

Заменим:

[annotations `enumerateObjectsUsingBlock:^(id<MKAnnotation> annotation..`

На:

[annotations `enumerate..Block:^(__unsafe_unretained id<MKAnnotation>..)`

Time Profiler:

Добавление: ~370ms

Затя №2

Заменим:

```
[annotations enumerate..Block:^(__unsafe_unretained id<MKAnnotation>..)
```

На:

```
for (id<MKAnnotation> annotation in annotations)
```


Затя №2

Заменим:

```
[annotations enumerate..Block:^(__unsafe_unretained id<MKAnnotation>..)
```

На:

```
for (id<MKAnnotation> annotation in annotations)
```

Time Profiler:

Добавление: ~348ms

Что происходит 348ms?

355.0ms	8.0%	118,0		▼-[MIMapIndex addAnnotations:] Demo
235.0ms	5.3%	126,0		▼MIQuadTreeInsertPoint Demo
81.0ms	1.8%	5,0		▼_MIQuadTreeSubdivide Demo
76.0ms	1.7%	76,0		_MIQuadTreeCreate Demo
28.0ms	0.6%	28,0		MIPointListCreate Demo

- ~230ms вставка в С-Дерево. Из них:
 - ~78ms - разбиение дерева
 - ~28ms - выделение памяти для списка

Что происходит 230ms?

```
void MIQuadTreeInsertPoint(MIQuadTreeRef tree, MIPoint point)
{
    if (!MKMapRectContainsPoint(tree->rect, (MKMapPoint){point.x, point.y})) return;

    ....// обновляем tree->count и центроид

    if (1 < (tree->count) && tree->level < 21)
    {
        _MIQuadTreeSubdivide(tree);

        ....// “спускаем” точки вершины в соответствующие листья
    }
    else
    {
        tree->pointList = MIPointListCreate(point, tree->pointList);
    }
}
```

Что происходит 230ms?

```
void MIQuadTreeInsertPoint(MIQuadTreeRef tree, MIPoint point)
{
```

```
    if (!MKMapRectContainsPoint(tree->rect, (MKMapPoint){point.x, point.y})) return;    26.8%
```

```
    ....// обновляем tree->count и центроид
```

```
    if (1 < (tree->count) && tree->level < 21)
    {
```

```
        _MIQuadTreeSubdivide(tree);    34.5%
```

```
        ....// “спускаем” точки вершины в соответствующие листья
```

```
    }
    else
    {
```

```
        tree->pointList = MIPointListCreate(point, tree->pointList);    12.5%
```

```
    }
}
```


Затя №3

Выделим метод с проверкой и метод с реализацией:

```
void _MIQuadTreeInsertPoint(MIQuadTreeRef tree, MIPoint point)
{
    // реализация вставки
}

void MIQuadTreeInsertPoint(MIQuadTreeRef tree, MIPoint point)
{
    if (!MKMapRectContainsPoint(tree->rect, (MKMapPoint){point.x, point.y})) return;

    _MIQuadTreeInsertPoint(MIQuadTreeRef tree, MIPoint point)
}
```

Затя №3

Выделим метод с проверкой и метод с реализацией:

```
void _MIQuadTreeInsertPoint(MIQuadTreeRef tree, MIPoint point)
{
    // реализация вставки
}

void MIQuadTreeInsertPoint(MIQuadTreeRef tree, MIPoint point)
{
    if (!MKMapRectContainsPoint(tree->rect, (MKMapPoint){point.x, point.y})) return;

    _MIQuadTreeInsertPoint(MIQuadTreeRef tree, MIPoint point)
}
```

Time Profiler:
Добавление: ~289ms

Затя №4

А что там с разбиением дерева?

```
MIQuadTreeRef _MIQuadTreeCreate(MIQuadTreeRef root, MKMapRect rect, unsigned int level)
{
    MIQuadTreeRef tree = malloc(sizeof(struct MIQuadTree));
    .... // обнуление полей структуры
}

void _MIQuadTreeSubdivide(MIQuadTreeRef tree)
{
    unsigned int level = tree->level + 1;

    tree->topLeftLeaf = _MIQuadTreeCreate(tree, _MIQuadTreeLeafRect(tree, 0), level);
    tree->topRightLeaf = _MIQuadTreeCreate(tree, _MIQuadTreeLeafRect(tree, 1), level);
    tree->bottomLeftLeaf = _MIQuadTreeCreate(tree, _MIQuadTreeLeafRect(tree, 2), level);
    tree->bottomRightLeaf = _MIQuadTreeCreate(tree, _MIQuadTreeLeafRect(tree, 3), level);
}
```

Затя №4

А что там с разбиением дерева?

```
MIQuadTreeRef _MIQuadTreeCreate(MIQuadTreeRef root, MKMapRect rect, unsigned int level)
{
    MIQuadTreeRef tree = malloc(sizeof(struct MIQuadTree));
    ....// обнуление полей структуры
}
```

```
void _MIQuadTreeSubdivide(MIQuadTreeRef tree)
{
    unsigned int level = tree->level + 1;
```

tree->topLeftLeaf = _MIQuadTreeCreate(tree, _MIQuadTreeLeafRect(tree, 0), level);	24.7%
tree->topRightLeaf = _MIQuadTreeCreate(tree, _MIQuadTreeLeafRect(tree, 1), level);	28.2%
tree->bottomLeftLeaf = _MIQuadTreeCreate(tree, _MIQuadTreeLeafRect(tree, 2), level);	24.7%
tree->bottomRightLeaf = _MIQuadTreeCreate(tree, _MIQuadTreeLeafRect(tree, 3), level);	20.0%

```
}
```


Затея №4

Заменим на инициализатор по адресу:

```
MIQuadTreeRef _MIQuadTreeCreate(MIQuadTreeRef tree, MIQuadTreeRef root, ....  
{  
    ...// инициализация необходимых значений  
    return tree;  
}
```

Затя №4

Заменим на инициализатор по адресу:

```
MIQuadTreeRef _MIQuadTreeCreate(MIQuadTreeRef tree, MIQuadTreeRef root, ....  
{  
    ... // инициализация необходимых значений  
    return tree;  
}
```

И 1 calloc вместо 4 malloc

```
void _MIQuadTreeSubdivide(MIQuadTreeRef tree)  
{  
    void *leaves = calloc(4, sizeof(struct MIQuadTree));  
  
    unsigned int level = tree->level + 1;  
  
    tree->topLeftLeaf = _MIQuadTreeCreate(leaves, tree, _MIQuadTreeLeafRect(tree, 0), level);  
    tree->topRightLeaf = _MIQuadTreeCreate(leaves + sizeof(struct MIQuadTree), ...  
    tree->bottomLeftLeaf = _MIQuadTreeCreate(leaves + sizeof(struct MIQuadTree) * 2, ...  
    tree->bottomRightLeaf = _MIQuadTreeCreate(leaves + sizeof(struct MIQuadTree) * 3, ...  
}
```

Затяжка №4

TimeProfiler: инициализатор по адресу:

Добавление: ~269ms

```
MIQuadTreeRef _MIQuadTreeCreate(MIQuadTreeRef tree, MIQuadTreeRef root, ....  
{  
    ... // инициализация необходимых значений  
    return tree;  
}
```

(malloc + memset даёт аналогичное время)

И 1 calloc вместо 4 malloc

```
void _MIQuadTreeSubdivide(MIQuadTreeRef tree)  
{  
    void *leaves = calloc(4, sizeof(struct MIQuadTree));  
  
    unsigned int level = tree->level + 1;  
  
    tree->topLeftLeaf = _MIQuadTreeCreate(leaves, tree, _MIQuadTreeLeafRect(tree, 0), level);  
    tree->topRightLeaf = _MIQuadTreeCreate(leaves + sizeof(struct MIQuadTree), ...  
    tree->bottomLeftLeaf = _MIQuadTreeCreate(leaves + sizeof(struct MIQuadTree) * 2, ...  
    tree->bottomRightLeaf = _MIQuadTreeCreate(leaves + sizeof(struct MIQuadTree) * 3, ...  
}
```

Демо

ВЫВОДЫ

- Алгоритмы и структуры данных - не только для печати в книгах :)
- Уважайте malloc и зовите его только в крайнем случае
- Instruments - наши друзья
- Измеряйте, изменяйте, измеряйте,

Что ещё?

- Structure layout & CPU cache
- References locality (temporal, spacial)
- Quadtree в виде массива

ЕСЛИ ВДРУГ

Интересное:

<http://en.wikipedia.org/wiki/Quadtree>

<http://www.akkadia.org/drepper/cpumemory.pdf>

<http://lsproengine.com/?p=530>

<http://stackoverflow.com/questions/892767/c-optimizing-member-variable-order/894032#894032>

Вопросы, предложения: shemet.dmitriy@gmail.com

Проект: https://github.com/poteryaysya/map_index

Спасибо за внимание